LETTER COMBINATORICS(LC) PROGRAM:

AN INTERFACE DESIGN AND UML PROJECT DEVELOPMENT.

(UNIFIED MODELING LANGUAGE-METADATA)

FRANK K. APPIAH[1]

BRITISH ACADEMY

COLLEGE OF SOFTWARE SYSTEM ENGINEERING

LONDON, UK.

(Software Modeling

and

System Modeling)

1 Dr. Frank Appiah hold studentships at the Bsc. (Bachelor in computer engineering, Kwame Nkrumah University of Science and Technology) in 2008, Msc.(software engineering, King's College London) in 2009/10 and 2 doctorates of philosophy (DPhil / PhD) in computer engineering from Kwame Nkrumah University of Science and Technology in 2013/14. He did his doctor of philosophy in computer science and engineering at King's College London in 2010/12 and postgraduate studies at the same place. He is a member of King's Alumni Group of Engineering, British Academy and Royal Society.

This is my full membership work of Bristish Academy (College level) in its first publication at the above institution. Any omitting or errors are mine only and can be communicated to when needed.

ABSTRACT

Letter combinatorics(LC) project is about interface diagrams and LC interfaces metadata as unified modeling language used as design content. It is a markup structure reversed engineered from computer programs developed by a Java programming language using Netbeans UML[4] tool. This depicts the interfaces diagrams and shows the extensible markup sourcefile artifacts in the interface system model.

1-0 INTRODUCTION

This document is about system models from interfaces design projected as unified modeling language for LCProgram. Letter Combinatorics(LC)[3] have the following requirements:

1. A countable number of sentences or phrases.
2. Counting the size of selected phrases for likely occurrence of subset equality of letters is called *count*.

3. A sentence with the number of letters specified is called Π. The logical structure of arithmetic such as +, - and = should be applied.
4. Discrete operations must include count, addition, subtraction and sizes.
5. The sizes of selected phrases are enumerated.
6. Proofs with the discrete operations on which the enumeration of sizes stops.
7. There is a possibility of summation equal to Π.

These combinatoric requirements are transformed into interfaces requirement analysis to yield the interfaces design process with focus on XMI metadata.

2-0 INTERFACES DESIGN

I will depict the wings of interface diagrams[4] as the interface design process. The interfaces to be look at includes the following:

- ILCProblem
- ILCProgram
- ISentence
- IMetaOperation
- IAlphanumeric
- IAlphaLabel

Now, it is the pictorial views of the interface diagrams of Letter combinatorics program, LCProgram.

2-1 A SINGLE INTERFACE DIAGRAM

A single interface diagram for the program design is shown in four different layouts namely incremental, hierarchical, symmetrical and orthogonal implementations.

Incremental Implementation

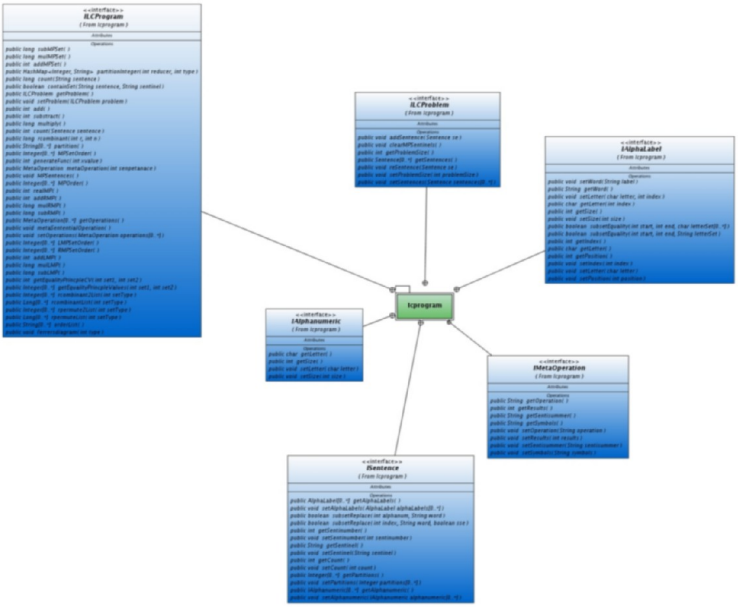

Illustration 1: Incremental Interface Diagram

Hierarchical Implementation

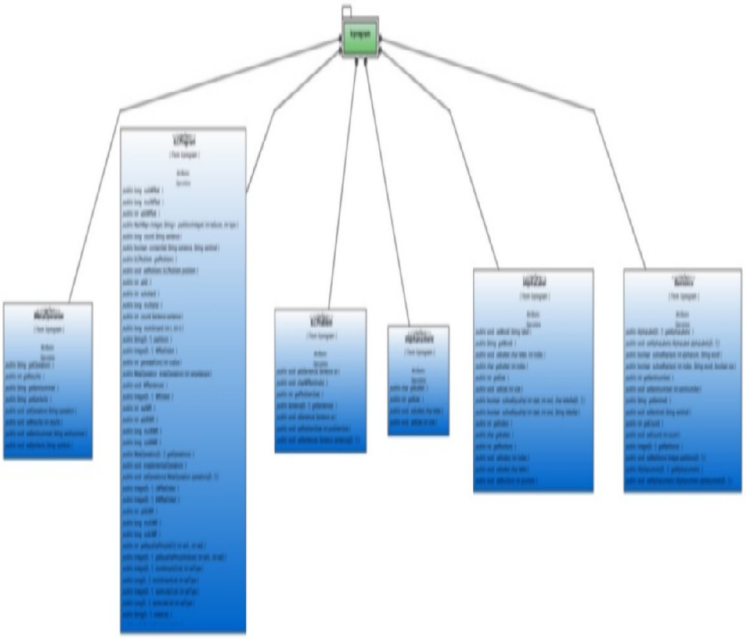

Illustration 2: Hierarchical Interface Diagram

Symmetrical Implementation

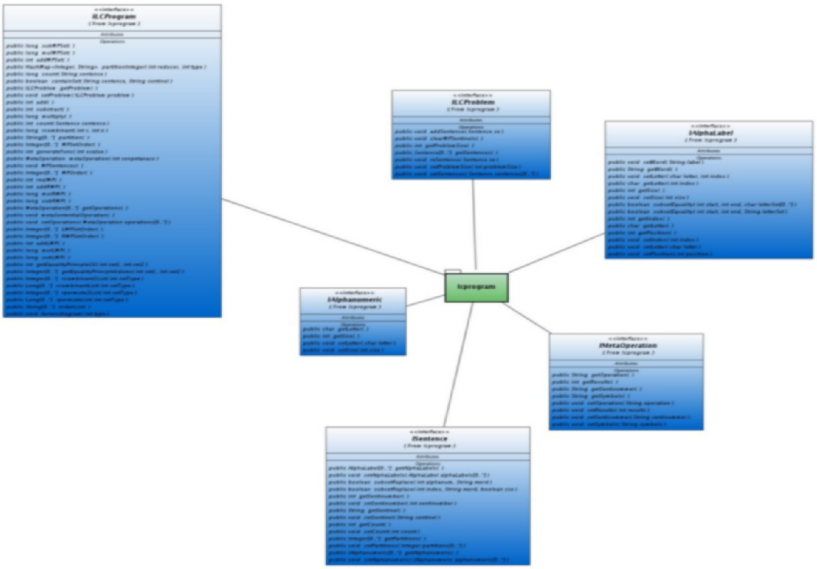

Illustration 3: Symmetrical Interface Diagram

Orthogonal implementations

Illustration 4: Orthogonal Interface Diagram

ILCProblem Interface Diagram

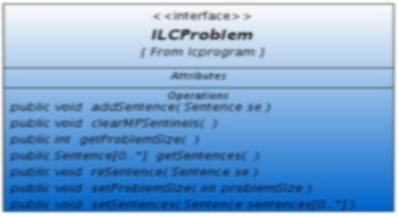

Illustration 5: ILCProblem Interface Diagram

ILCProgram Interface Diagram

Illustration 6: ILCProgram Interface Diagram

ISentence Interface Diagram

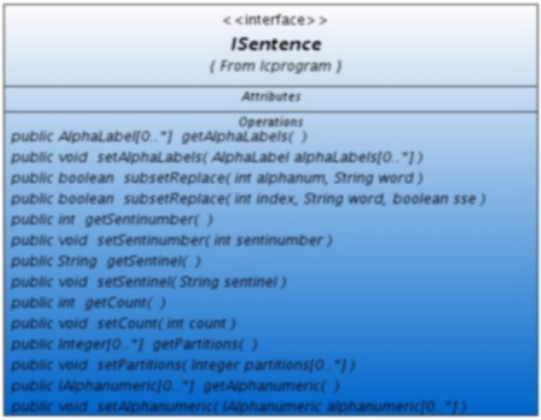

Illustration 7: ISentence Interface Diagram

IMetaOperation Interface Diagram

Illustration 8: IMetaOperation Interface Diagram

IAlphanumeric Interface Diagram

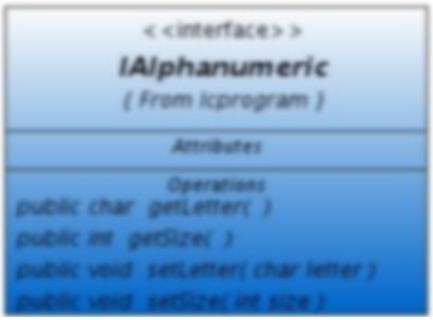

Illustration 9: IAlphanumeric Interface Diagram

IAlphaLabel Interface Diagram

Illustration 10: IAlphaLabel Interface Diagram

3-0 UML SOURCEFILE ARTIFACTS

Extensible Markup Interchange (XMI-UML) System[5] is a structure of unified modeling languages defined in extensible markup language. This is a project from Object Management Group at http://omg.org/UML/1.4. The version of UML to be described here is 2. UML system model is a metamodel. UML system model is a model about stereotype models. There are three stereotypes in UML model namely:

1. Datatype
2. Folder and
3. Interface.

Much of concern here is the Interface stereotype. XMI-UML header is shown below:

Illustration 11: UML Header

This section is about Java[2] Interface languages used in the interfaces development process. The interface diagram above is now implemented shown as several source files artifacts and source code blocks. The Java interfaces to be look at are as follows:

1. ILCPROBLEM JAVA INTERFACE
2. ILCPROGRAM JAVA INTERFACE
3. ISENTENCE JAVA INTERFACE
4. IMETAOPERATION JAVA INTERFACE
5. IALPHALABEL JAVA INTERFACE and
6. IALPHANUMERIC JAVA INTERFACE.

3-1 INTERFACE SOURCEFILES

UML ILCPROBLEM SourceFile Artifact

```
<UML:SourceFileArtifact sourcefile="{${src.dir}}/
lcprogram/ILCProblem.java" xmi.id="DCE.368FC832-3E16-
F06B-F6C9-61F0E3EC977D" owner="DCE.63DB49F8-3C8D-
5D28-54E0-0BEB9D43A694"/>
```

```
<UML:TaggedValue xmi.id="DCE.E6DE1290-248C-1A39-FB0A-
3C9CD22F32F8" name="documentation">
```

Illustration 12: UML Interface ILCPROBLEM

Check Appendix for UML Interface ILCPROBLEM at Line 123.

ILCPROBLEM JAVA INTERFACE

```
1.  public interface ILCProblem {
2.
3.      void addSentence(Sentence se);
4.
5.      void clearMPSentinels();
6.
7.      int getProblemSize();
8.
9.      ArrayList<Sentence> getSentences();
10.
11.     void reSentence(Sentence se);
12.
13.     void setProblemSize(int problemSize);
14.
15.     void setSentences(ArrayList<Sentence>
   sentences);
16.
17.}
```

UML ILCPROGRAMSourceFile Artifact

```
<UML:SourceFileArtifact
sourcefile="{${src.dir}}/lcprogram/ILCProgram.java"
xmi.id="DCE.2548D966-DC38-528F-BB7A-FBC7D876BB4F"
owner="DCE.DCCBE6AD-47A9-2C95-6134-8A8AEA6B81DF"/>
```

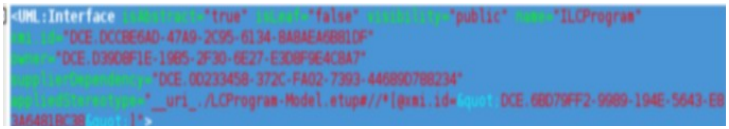

Illustration 13: UML Interface ILCPROGRAM

Check Appendix for UML Interface ILCPROGRAM at Line 339.

ILCPROGRAM JAVA INTERFACE

```
18. public interface ILCProgram {
19.
20.     public long subMPSet();
21.
22.     public long mulMPSet();
23.
24.     public int addMPSet();
25.
26.     public HashMap<Integer, String>
```

```
   partitionInteger(int reducer, int type);
27.
28.     public long count(String sentence);
29.
30.     public boolean containSet(String sentence,
   String sentinel);
31.
32.     public ILCProblem getProblem();
33.
34.     public void setProblem(ILCProblem problem);
35.
36.     public int add();
37.
38.     public int substract();
39.
40.     public long multiply();
41.
42.     public int count(Sentence sentence);
43.
44.     public long rcombinant(int r, int n);
45.
46.     public List<String> partition();
47.
48.     public List<Integer> MPSetOrder();
49.
50.     public int generateFunc(int xvalue);
51.
52.     public MetaOperation metaOperation(int
   senpetanace);
53.
54.     public void MPSentences();
55.
56.     public List<Integer> MPOrder();
57.
58.     public int realMP();
59.
60.     public int addRMP();
61.
```

```
62.     public long mulRMP();
63.
64.     public long subRMP();
65.
66.     public List<MetaOperation> getOperations();
67.
68.     public void metaSententialOperation();
69.
70.     public void setOperations(List<MetaOperation>
   operations);
71.
72.     public List<Integer> LMPSetOrder();
73.
74.     public List<Integer> RMPSetOrder();
75.
76.     public int addLMP();
77.
78.     public long mulLMP();
79.
80.     public long subLMP();
81.
82.     public int getEqualityPrincpleCV(int set1, int
   set2);
83.
84.     public List<Integer> getEqualityPrincpleValues(
   int set1, int set2);
85.
86.     public List<Integer> rcombinant2List(int
   setType);
87.
88.     public List<Long> rcombinantList(int setType);
89.
90.     public List<Integer> rpermute2List(int
   setType);
91.
92.     public List<Long> rpermuteList(int setType);
93.
94.     /**
```

```
95.      * It results to count * e^ (count/limitsize) *
   numberOrderLimit.
96.      * The count is the enumeration of a sentence,
   limitsize=20
97.      * and numberOrderLimit has a value for 20!.
98.      * @return It returns a possible number of
   order
99.      */
100.    public List<String> orderList();
101.
102.    /**
103.     * It prints ferrers diagram for MP, MPSet,
   RMPSet and LMPSet partitions.
104.     * @param type Type of Set.
105.     */
106.    public void ferrersdiagram(int type);
107.
108.}
```

UML ISENTENCE SourceFile Artifact

```
<UML:SourceFileArtifact
sourcefile="{${src.dir}}/lcprogram/ISentence.java"
xmi.id="DCE.81846E60-05EA-B89B-C493-E83E5D745BB2"
owner="DCE.443403AB-C4CD-0013-4497-27119B267A49"/>
```

Check Appendix for UML Interface ISENTENCE at Line 159.

Illustration 14: UML Interface ISENTENCE

ISENTENCE JAVA INTERFACE

```
109.public interface ISentence {
110.
111.     /**
112.      * It is a word from a sentence with all its
   alphabet labelling
113.      * structure.
114.      * @return List of AlphaLabels.
115.      */
116.     public ArrayList<AlphaLabel> getAlphaLabels();
117.
118.     /**
119.      * It sets alphalabels of a sentence's word.
120.      * @param alphaLabels List of AlphaLabels.
121.      */
122.     public void
   setAlphaLabels(ArrayList<AlphaLabel> alphaLabels);
123.
124.     /**
125.      * It replaces a word in an alphalabel with a
   word if the number
126.      * of letters are equal or not to that of the
   replaced word.
127.      * @param alphanum number of sentence
   alphalabel
128.      * @param word word used in replacement.
129.      * @return If successfully replaced returns
```

```
   true or false.
130.      */
131.     public boolean subsetReplace(int alphanum,
   String word);
132.
133.     /**
134.      * It replaces a word in an alphalabel with a
   word with a concantenation
135.      * of letters are equal to that of the
   original word plus the append word.
136.      * @param index index in a sentence.
137.      * @param word  word used in appendment.
138.      * @param sse   whether to check for append
   after or before.(true=after, false=before)
139.      * @return If successfully replaced returns
   true or false.
140.      */
141.     public boolean subsetReplace(int index, String
   word, boolean sse);
142.
143.     /**
144.      * It returns the number of this sentence.
145.      * @return Number of sentence.
146.      */
147.     public int getSentinumber();
148.
149.     /**
150.      * It sets the sentinel number of a sentence.
151.      * @param sentinumber A number of a sentence.
152.      */
153.     public void setSentinumber(int sentinumber);
154.
155.     String getSentinel();
156.
157.     void setSentinel(String sentinel);
158.
159.     int getCount();
160.
```

```
161.     void setCount(int count);
162.
163.     List<Integer> getPartitions();
164.
165.     void setPartitions(List<Integer> partitions);
166.
167.       /**
168.        * It returns the alphanumeric labelling of a
   word.
169.        * @return List of structures made up of an
   alphabet with index.
170.        */
171.     public List<IAlphanumeric> getAlphanumeric();
172.
173.     /**
174.        * It sets the alphanumeric labelling of a
   word.
175.        * @param alphanumeric List of alphanumerics
176.        */
177.     public void setAlphanumeric(List<IAlphanumeric>
   alphanumeric);
178.
179.}
```

UML IMETAOPERATION SourceFile Artifact

```
<UML:SourceFileArtifact
sourcefile="{${src.dir}}/lcprogram/IMetaOperation.jav
a" xmi.id="DCE.F2F01A6F-929A-252A-AD6E-DECFCBAC9504"
owner="DCE.04600453-1E07-2170-80BA-523E4B4EB73E"/>
```

Illustration 15: UML Interface IMETAOPERATION

Check Appendix for UML Interface IMETAOPERATION at Line 16.

IMETAOPERATION JAVA INTERFACE

```
180.public interface IMetaOperation {
181.
182.    String getOperation();
183.
184.    int getResults();
185.
186.    String getSentisummer();
187.
188.    String getSymbols();
189.
190.    void setOperation(String operation);
191.
192.    void setResults(int results);
193.
194.    void setSentisummer(String sentisummer);
195.
196.    void setSymbols(String symbols);
197.
198.}
```

UML IALPHALABEL SourceFile Artifact

```
<UML:SourceFileArtifact
sourcefile="{${src.dir}}/lcprogram/IAlphaLabel.java"
xmi.id="DCE.5CDD3984-14EA-D69E-08F1-9A1EF15BF37D"
owner="DCE.5B566B69-2A41-21C2-509A-259FDE9EA24E"/>
```

Illustration 16: UML Interface IALPHALABEL

Check Appendix for UML Interface IALPHALABEL at Line 248.

IALPHALABEL JAVA INTERFACE

```
199.public interface IAlphaLabel {
200.
201.     /**
202.      *  It sets a word for labelling
203.      * @param label The label of word from a
   sentence.
204.      */
205.     public void setWord(String label);
206.
```

```
207.     /**
208.      * It returns the label word of a
   sentence.
209.      * @return A label word made of alphabets
210.      */
211.     public String getWord();
212.
213.     /**
214.      * It sets  a new character at the
   position, index of a  word.
215.      * @param letter Character used in
   replacement.
216.      * @param index Position in word to do
   the replacement.
217.      */
218.     public void setLetter(char letter, int
   index);
219.
220.     /**
221.      * It returns the character or letter at
   the specified index.
222.      * @param index position to make a char
   retrieval.
223.      * @return A specific character from a
   word.
224.      */
225.     public char getLetter(int index);
226.
227.     /**
228.      * It returns the size of words.
229.      * @return Size of words.
230.      */
231.     public int getSize();
232.
233.     /**
```

```
234.      * It sets the size of words.
235.      * @param size Sise of wordss to check to
   consistency.
236.      */
237.     public void setSize(int size);
238.
239.     /**
240.      * It checks for subset equality of word
   with a specific chararray letterset.
241.      * @param start Start position to find
   the first letter.
242.      * @param end end position to find the
   last letter.
243.      * @param letterSet An array of letters.
244.      * @return True if equal or false.
245.      */
246.     public boolean subsetEquality(int start,
   int end, char[] letterSet);
247.
248.     /**
249.      * It checks for subset equality of word
   with a specific string letterset.
250.      * @param start start Start position to
   find the first letter.
251.      * @param end  end end position to find
   the last letter.
252.      * @param letterSet  letterSet An array
   of letters.
253.      * @return True if equal or false.
254.      */
255.     boolean subsetEquality(int start, int
   end, String letterSet);
256.
257.     public int getIndex();
258.
```

```
259.     public char getLetter();
260.
261.     /**
262.      *  It returns the position of word in a
   sentence
263.      *
264.      * @return Position of sentence.
265.      */
266.     public int getPosition();
267.
268.     /**
269.      *
270.      * @param index
271.      */
272.     public void setIndex(int index);
273.
274.     public void setLetter(char letter);
275.
276.     /**
277.      * It sets the position of word in a
   sentence.
278.      * @param position Position in sentence
279.      */
280.     public void setPosition(int position);
281.}
```

UML IALPHANUMERIC SourceFile Artifact

```
<UML:SourceFileArtifact
sourcefile="{${src.dir}}/lcprogram/IAlphanumeric.java
" xmi.id="DCE.5E0A7EC7-8B1C-1CC4-7391-6A110B676EB4"
```

```
owner="DCE.6E3531A0-6B56-DDC4-AEF6-E77EE1DB68A5"/>
```

Illustration 17: UML Interface IALPHANUMERIC

Check Appendix for UML Interface IALPHANUMERIC at Line 144.

IALPHANUMERIC JAVA INTERFACE

```
282.public interface IAlphanumeric {
283.
284.    char getLetter();
285.
286.    int getSize();
287.
288.    void setLetter(char letter);
289.
290.    void setSize(int size);
291.
292.}
```

4-0 SUMMARY

This is a document on interfaces design, interfaces project and UML interfaces development on LCProgram, which is used in the computing of combinatorial values or algorithms.

The document provides a detailed interface diagrams on the six interfaces of the program. A hierarchical, incremental, symmetrical and orthogonal implementations of the interface diagram was also depicted.

A single interface diagram of each Java Interface is provided by the Netebeans UML tool. In total about 10 pictorial interface diagrams are depicted in the design methodology process. A *unified project development* is carried out in system modeling the interfaces that outputted XMI metadata as an interchange from the UML diagramming process.

REFERENCES

1. Nell Dale and Chip Weens(1994): Pascal, 4 Edition, D. C. Heath and Company.ISBN: 0-669-34219-X.
2. Java SDK(2019), Oracle Inc, http://www.oracle.com. ((Accessed August, 2019).
3. Frank Appiah(2019), Letter Combinatorics on Marriage Problem, Teaching Document, Institution Above.
4. Netbeans (2019), Netbeans UML Tool, http://www.netbeans.org.
5. OMG(2019), Object Management Group, http://omg.org/UML/1.4.

ILLUSTRATION INDEX

INDEX

APPENDIX A LCPROGRAM-UNIFIED MODELING LANGUAGE: XML METADATA INTERCHANGE. (xmi-uml)